命名数据网络中分组报文缓存优化方法 *

智　江 [1,2]，李　俊 [1]，吴海博 [1]

(1. 中国科学院计算机网络信息中心，北京 100190; 2. 中国科学院大学，北京 100190)

摘　要：传统网络缓存系统中数据包级别的缓存难以实现，信息中心网络的出现使这个难题得以缓解。即使如此，数据包级别的缓存仍然面临严重的扩展性问题。通过分析当前限制数据包级别缓存实现的若干问题，提出了一种分组报文缓存优化方法。这种方法通过根据分组前缀而非单个报文前缀建立索引来减少高速存储器的使用量，同时分组级别的流行度也用于优化缓存决策。定义了大量的评估指标，并通过广泛的实验来评估此方案的性能。实验结果表明，与之前的数据包级别的缓存方案相比，此方法可以大大减少高速存储器使用量，并且在服务器负载减少率、平均跳数减少率和平均缓存命中率方面取得显着改善。

关键词：包缓存；命名数据网络；网内缓存；缓存决策

中图分类号：TP393.0　　doi: 10.3969/j.issn.1001-3695.2018.03.0177

Grouped-packet caching optimization approach for NDN

Zhi Jiang[1,2], Li Jun[1], Wu Haibo[1]

(1. *Computer Network Information Center*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*; 2. *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

Abstract: Packet-level caching is difficult to implement in the traditional caching system. The emergence of Information-centric networking has alleviated this problem. However, the packet-level caching is still facing severe scalability issues. This paper analyzed the issues which limit the implementation of packet-level caching and proposed a packet-level caching optimization approach. This scheme reduces the consumption of fast memory by creating the index with group prefixes instead of the packet prefixes, while the group-level popularity is also used to optimize caching decision. This paper evaluated the performance of the scheme through extensive simulation experiments regarding a wide range of performance metrics. The experimental results indicate the scheme can reduce the fast memory usage and achieve significant improvement in terms of server load reduction ratio, average hop reduction ratio and average cache hit ratio, compared with current packet-level caching schemes.

Key words: packet caching; named data networking (NDN); in-network caching; caching decision

0　引言

现今互联网的主流应用已经由端到端模式的通信转变为了以内容分发和获取为主的应用。根据思科 VNI 报告中错误!未找到引用源。预测，预计 2021 年内容获取类应用产生的网络流量（如互联网视频、Web 数据、文件共享等）将占总网络流量的 85%以上。网络缓存这是缓解上述流量压力的重要技术之一。网络缓存设备可以暂存内容对象并响应之后的用户请求。一些研究指出，相比于整个内容对象的缓存，更细的粒度的缓存（比如数据包级别的缓存）能够获得更好的缓存效果。因为数据包级别的缓存可以做出更细粒度的缓存决策，从而可以进一步提高网络性能，尤其是当对象的不同部分具有不同的流行度时错误!未找到引用源。然而，在传统的 TCP/IP 网络中，数据包级别的缓存却难以实现，因为它需要借助诸如 DPI 等会产生较高开销的技术，这将造成严重性能瓶颈和扩展性问题。

传统缓存中数据包级别的缓存所面临的问题可以通过新近出现的信息中心网络架构（information-centric networking，ICN）来缓解[3~6]。ICN 架构采用了接受者驱动的通信模型，并且将命名的数据作为网络中传输的基本元素。通过名字可以标识这些数据包，因此在 ICN 体系结构中，命名的数据可以存储在任何有缓存能力的网络设备中（如路由器）。当请求消息到达时，网络设备可以从其存储库中检索内容数据。如果内容已被缓存，则可以直接响应用户请求，而无需将请求转发向内容服务器。这样，网络带宽消耗和内容服务器的负载可以显着降低。同时，

收稿日期：2018-03-13；修回日期：2018-04-24　　基金项目：国家自然科学基金资助项目（61672490，61602436，61601443）；国家重点研发计划资助项目（2017YFB1401500）

作者简介：智江（1983-），男，山西太原人，博士研究生，主要研究方向为未来互联网技术、网络缓存等（zhijiang@cstnet.cn）；李俊（1968-），男，研究员，博士，主要研究方向为未来互联网技术、网络安全等；吴海博（1981-），男，助理研究员，博士，主要研究方向为未来互联网、缓存技术、对等网络.

由于请求延迟的减少，用户体验也将大大改善。

ICN 架构的出现缓解了数据包级别缓存遇到的一些问题，然而，ICN 架构下数据包级别的缓存在实现上仍然存在诸多挑战：高速存储器（如 SRAM）的容量限制;对于线速转发的要求;数据包级别的流行度统计等。

为了以较低的成本实现数据包级别的缓存，本文根据空间局部性原理，引入一种了基于分组数据包的缓存方法（grouped-packet caching，GPC）。在 GPC 中，我们使用分组前缀代替单个报文的前缀来建立索引项目以减少高速存储器容量的消耗。另一方面我们利用一个 Bloom Filter 的存储器来过滤无效缓存查询请求，从而加速请求转发。此外，我们使用分组的流行度来优化缓存决策。具体来说，本文的主要贡献总结如下：

a)本文分析了限制数据包级别缓存实现的相关问题。并且基于空间局部性原理，引入分组报文的概念来缓解数据包级别缓存中存在的扩展性问题。

b)本文提出了一种名为 GPC 的数据包级别的缓存机制，可以显著加速那些未缓存的数据包的转发过程，同时大大降低高速存储器的使用量。

c)定义了若干评价指标来评估 GPC 机制的效果。实验结果表明，高速存储器的使用量显着降低。同时，与主流的几种包级别的缓存方案相比，GPC 显示出了更优的缓存性能。

1　相关工作

在传统 TCP/IP 网络中，IP 数据包由五元组标识，而其有效负载对网络设备不可见。为了实现数据包级别的网络缓存，则需要通过特定的方法（如 suppressing replicated data，delta coding 或深度包检测等）来消除数据冗余。由于计算开销成本过高，在传统的 TPC/IP 网络难以实现数据包级别的缓存。因此，传统网络缓存通常存储完整的数据对象而不是单独的数据包。

近来出现的信息中心网络（ICN）使数据包级别的缓存变得可行。ICN 体系结构将对象划分为命名的数据块，用户通过向网络发送一系列包含数据块名称的请求来获取内容对象。这些命名的数据块可以缓存在任何具有缓存能力的 ICN 设备中，这些设备也可以响应对已缓存数据块的请求。

由于 ICN 的出现，数据包级别的缓存方案引起了更多研究人员的关注。在文献**错误!未找到引用源。错误!未找到引用源。**中，作者通过确定内容缓存的位置来提高整体缓存效果。但是，它们没有考虑内容流行度的对缓存性能的影响，而事实上这一点是不能忽视的。最近，学者们普遍认识到内容流行对提高缓存性能起着至关重要的作用**错误!未找到引用源。**。因此，一些缓存策略通过同时考虑缓存位置和内容流行度来提高缓存效率[10-15]。然而，这些研究缺乏对对象流行度和数据包流行度之间的关系的分析，同时忽略了数据包级别流行度统计量的开销，这将会产生严重的扩展性问题。

在文献**错误!未找到引用源。**中，作者阐述了分块缓存的优势，并提出了一种量化效用的方法。评估结果表明，简单的基于效用的网内缓存算法和低复杂度的均匀分块足以发挥分块缓存的最大优势。Thomas 等人**错误!未找到引用源。**提出了一种面向内容对象的数据包级别的缓存机制（OPC）。在 OPC 中，缓存索引条目是基于整个对象而建立，而非针对单个的数据包建立（一个对象可以划分为多个数据包）。通过这种方法可以大大节省高速内存使用量。为了实现数据包级别的缓存，在这种机制中缓存系统中需要连续保存对象的前 n 个块而没有间隙，并且在缓存索引中加以标识。 OPC 可以解决数据包缓存的两个常见问题：循环替换和大对象污染问题。但是，由于作者没有考虑对象内部的流行度差异，因此缓存性能仍有提升空间。此外存储连续的 n 的数据块的方法不够灵活（分块缓存必须按序存储），造成做出的缓存决策也较为僵化。

在总结上述已有工作的基础上，提出了一种新的数据包级别缓存的优化方法。

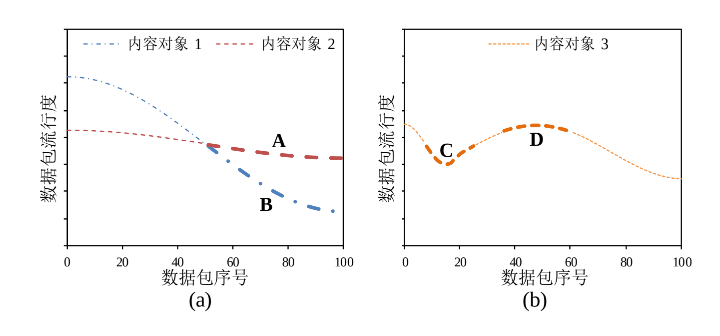


图 1 数据包流行度差异

2　问题分析

2.1　有限的存储资源

在一个网络缓存设备中，内容数据通常存放在 DRAM 或者 SSD 等低速存储器中，而查询索引则存储在 SRAM 等高速存储器中。在传统缓存中，一条索引对应一个整体的内容对象，而在包级别的缓存系统中，一条索引对应一个独立的数据包。由于一个内容对象通常可以划分为大量的数据包，相应所需的索引条目的数量也大大增多了。由此可知，与对象存储相比，数据包级别的缓存需要占用更多的高速内存来存储索引表，而高速存储器的成本限制了数据包级别缓存的发展。

2.2　无效的缓存查找

在 ICN 架构中，缓存系统不再是单个缓存设备，而是一个由大量设备组成的复杂的缓存网络。尽管缓存网络的总体容量可以很大，但由于受到线速转发的条件的制约，单个缓存节点的缓存容量大大受限。由于这个限制，单个缓存节点具有相对较低的缓存命中率。换句话说，大多数用户请求不会被单个缓存设备满足，而这些未命中的请求应该直接转发到下一跳。尽管如此，路由器总是查询外部存储器中的索引表，这将会导致性能瓶颈。

2.3　数据包级别流行度的难题

为了提高网络缓存系统的效率，将高热度的内容对象快速

推送到网络边缘至关重要。在传统的网络缓存系统中，缓存节点通常存储整个对象而不是数据包。实际上，一个内容对象的不同部分，特别是视频文件等，可能具有不同的流行度。因此，包级别的缓存的可以提高缓存性能错误!未找到引用源。。但是，基于对象级别的流行统计不能充分发挥数据包级别缓存的优势，因为使用对象的流行度统计来估计数据包的流行度通常是不准确的。我们在图 1 用一个简单例子来说明这个问题。在图 1（a）中，内容对象 1 的整体热度高于内容对象 2。然而，对于部分 A 和部分 B，可以看到，A 中的数据包比 B 中的数据包具有更高的热度。因此，基于内容对象的热度统计，A 的热度可能被低估，这可能会减少缓存这些数据包的几率。在图 1（b）中，C 的序号小于 D，但 D 的热度更高。在这种情况下，优先选择内容对象中排在前面的数据包的缓存方法（如 OPC **错误!未找到引用源。**）将过高估计 C 部分的热度，而决定缓存 C 而不是热度更高的 D。以上两种情况都由于存储了热度较低的内容而导致缓存性能的下降。由此可知数据包级别的热度统计对于提升缓存性能十分重要。但同时这也会带来过多系统的开销，因此阻碍了包级别缓存的发展。

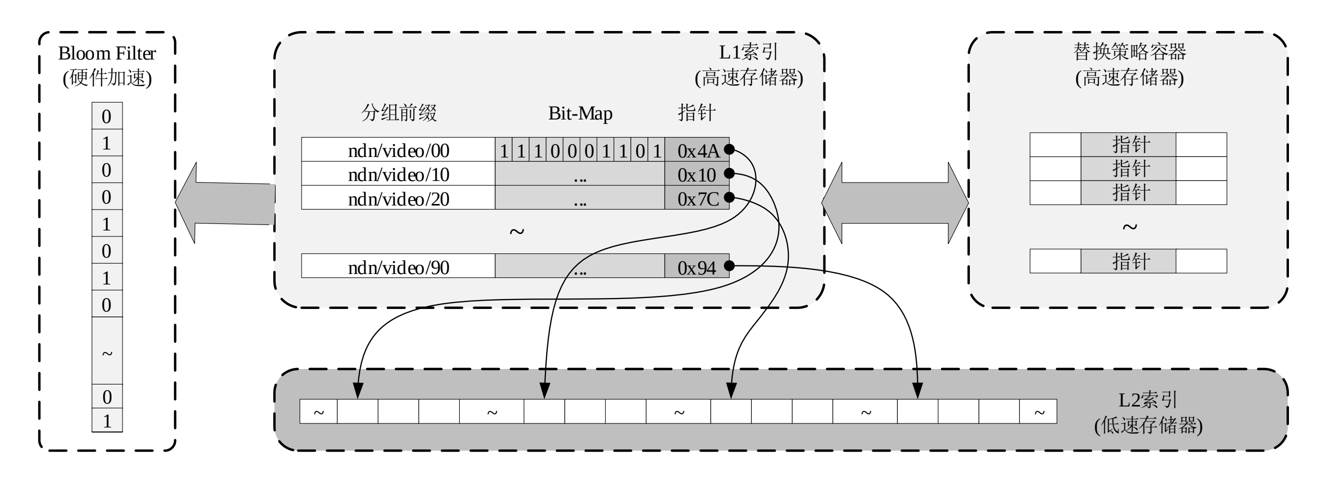


图 2 CS 索引数据结构

3　方案设计

3.1　系统模型

为了简化描述，本文以 ICN 体系结构中的典型代表 NDN（named data networking）作为示例来演示本文的缓存机制。

在本文中，认为一个 NDN 网络由多个具有有限容量缓存存储结构（content store，CS）的路由器组成。在 NDN 体系结构中，存在两种类型的数据包，即 Interest 报文和 Data 报文。每一个 Data 数据包都具有一个名字前缀，客户端发送包含该名字前缀的 Interest 数据包到网络中以获取相应的 Data 数据包。中间路由器根据存储在转发信息库（FIB）中的信息将请求逐跳转发至内容持有者。转发信息库通过基于名称的路由协议（如 OSPFN 错误!未找到引用源。或 NLSR 错误!未找到引用源。）建立。在请求过程中，路由器的 Pending Interest Table（PIT）结构中记录该请求过程的路径的信息，通过这些信息，Data 数据包可以沿着反向路径返回到请求者。

此外，假设 Data 数据包的大小等于网络的 MTU（默认为 1500 Byte）。同时 NDN 路由器中具有内容热度统计表（PST），可以记录和更新一段时间内数据包的访问热度。缓存的 Data 数据包存储在低速存储器中，例如 DRAM 或 SSD 等。而索引信息则存储在如 SRAM 的高速存储器中。NDN 中数据包的名字由对象前缀（Object_Prefix）和序号（Seq）组成，其中序号表示该数据包在内容对象中的顺序。例如，名字前缀 */ndn/file/video/sample/57* 表示对象 */ndn/file/video/sample* 的一个数据包，并且数据包的顺序号为 57。

3.2　缓存机制

3.2.1 报文分组

通常情况下，索引条目使用哈希表结构存储于高速存储器中。哈希表的优点是高效：元素操作（如查找，插入和删除）的时间复杂度为 O（1）。但是，与对象级别的缓存不同，报文级别的缓存中存储的最小单位是数据包。由于一个内容对象通常由至少数百个块组成，因此会生成大量索引条目，从而需要占用大量的高速内存空间来存储这些索引条目。但事实上，隶属于同一个对象的不同数据包，它们的对象前缀都相同，区别仅仅是最后的序号部分。因此，减少内存消耗的一个可能的解决方案是采用字典树结构来存储索引。然而，尽管字典树可以通过共享相同的对象前缀来节省存储空间，但字典树的查询效率相对较低，尤其是当前缀的数量较大时。例如，要搜索前缀 /ndn/file/video/sample/57，它需要进行五次匹配的操作。为了兼顾查询效率和存储空间节省，采用了分组报文索引的方式。

根据空间局部性原理：如果在某个特定的时间里某个特定数据块被请求，则相近的数据块也可能在近期被请求。基于上述观点，将 N 个连续数据包作为一个分组，并将该组中第一个数据包的名字前缀指定为分组前缀。缓存中的索引条目基于分组前缀来创建的。在每个索引条目中，添加一个 Bit-Map 结构来标识分组中的某个数据包是否已被缓存。当且仅当相应的位值为 1 时，表明该数据包被缓存；反之该值为 0 时，表示该数据包未被缓存。实际的 Data 数据包以 unorder_set 的形式被存

储在低速存储器中。unorder_set 是一种基于哈希表的容器，其检索的时间复杂度为 O（1）。索引条目包含一个指针，指向低速存储器中的该分组的容器。此外，高速存储器中还包含一个侵入式列表，用来对索引条目进行排序来完成替换功能。图 2 中描述了 GPC 策略使用的数据结构。缓存模块的核心操作包括有：

a)插入。路由器首先根据数据包隶属的分组的前缀创建索引条目（如果此条目此前未创建），然后将相应的 Bit-Map 的位值设置为 1。

b)换出。将 Bit-Map 中的相应位值设置为 0。此外，如果该索引条目的 Bit-Map 中所有值均为 0，则同时删除该索引条目。

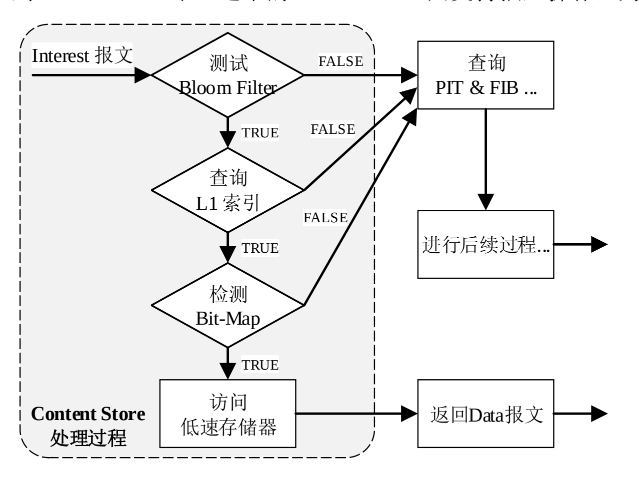
c)查找。根据数据包隶属的的分组前缀首先查找 L1 级索引。 如果找到分组前缀，则路由器检查 Bit-Map。如果对应位值为 1，则路由器访问低速存储器以检索块数据。否则直接转发请求。

3.2.2 请求过滤

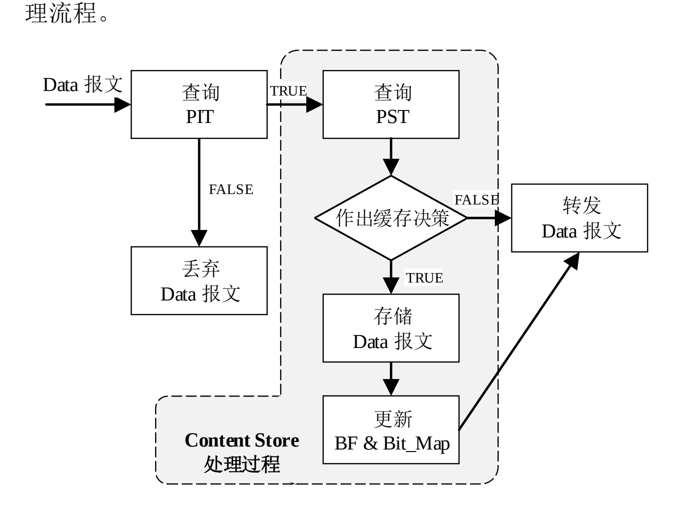
正如之前所讨论的，由于缓存容量的限制，单个路由器无法满足大部分的用户请求。为了加速数据包转发，我们利用一个芯片内置的 Bloom Filter 来减少外部存储器的访问次数。Bloom Filter 是一种查询效率很高并且占用空间较小的数据结构，可以快速而高效地对集合的内容进行检索。将索引信息同步到 Bloom Filter 中。通常的 Bloom Filter 只支持插入操作，为了增加对移除操作的支持我们采用的 Bloom Filter 的一种扩展结构 Counting Bloom Filter。路由器缓存的 L1 索引条目的各项操作都被同步到 Bloom Filter 中。由于存储在 L1 索引中的条目是基于分组前缀的，所以存储在 Bloom Filter 中的相应元素数量也大大减少了。例如，对于配备了 4GBytes DRAM 的路由器，当分组大小为 32 并且 Bloom Filter 的假阳性概率为 0.05 时，只需要 273KBytes 来存储分组前缀。而如果要存储所有的报文前缀，Bloom Filter 所需的容量大约是 8.5MBytes。显然，基于分组报文的缓存不但可以减少 L1 索引条目的数量也可以降低所需的 Bloom Filter 的容量。所需的 Count Bloom Filter 的容量可以通过式（1）计算。其中 $C_{storage}$ 表示低速存储器的容量，*packet_size* 表示数据包的大小，*group_size* 表示分组的大小，ε 表示假阳性的概率。

$$\frac{C_{Storage}\times\log_2 e\times\log_2(\frac{1}{\varepsilon})}{2\times packet_size\times group_size} \tag{1}$$

当用户请求到达时，路由器首先测试 Bloom Filter 以确定 L1 索引中是否存在该分组前缀。如果测试结果为真，则路由器继续访问外部高速存储器以完成后续操作。否则，路由器直接转发请求。这样，绝大多数的用户请求将被过滤，并且访问外部存储器的次数大大减少。图 3（a）描述了 Interest 报文的处理流程。



(a) Interest 报文处理过程



(b) Data 报文处理过程

图 3 数据包处理流程

3.2.3 分组级别的热度统计

数据包级别的热度统计很难实现，因为它将产生巨大的开销，尤其是是存储空间的消耗。因此，我们提出了分组级别的热度统计的概念。根据空间局部性原理可以推知，顺序上临近的报文具有相似的请求热度。因此，我们使用分组热度代替数据包热度来辅助路由器做出缓存决策。路由器在 PST 中记录分组热度信息，并根据分组前缀创建索引。这样将大大减少热度统计带来的开销。

当 Data 数据包到达时，路由器查找 PST 以获取该数据包隶属的分组的热度。如果它的热度值大于一个的阈值，则路由器决定存储该 Data 数据包。否则，路由器只转发 Data 数据包而不进行缓存。该阈值定义如下：

$$threshold=\theta\times Min(popularity)\quad \theta\in[0,1] \tag{2}$$

其中 *Min（popularity）*表示当前存储在 CS 中的内容的热度的最小值。θ 是一个调整参数，用来适应网络中内容对象热度的动态变化，较小的 θ 值适合于网络内容热度变化较快的场景。图 3（b）描述了 Data 数据包处理的流程。

4 性能评估

4.1 实验设定

为了评估本方案的实际性能，本文在 ndnSIM 错误!未找到引用源。

上实现了该方案。ndnSIM 是一个基于 NS-3 错误!未找到引用源。实现的模块，加入了 NDN 特性的数据结构和转发逻辑。基于 ndnSIM 增加了一些数据结构并修改了部分转发逻辑来实现 GPC 方案。在实验评估里，定义了三种不同分组大小的方案，其大小分别为 8、16 和 32，命名为 GPC-8，GPC-16 和 GPC-32。本文将这三种 GPC 策略与如下包级别的缓存策略进行对比：

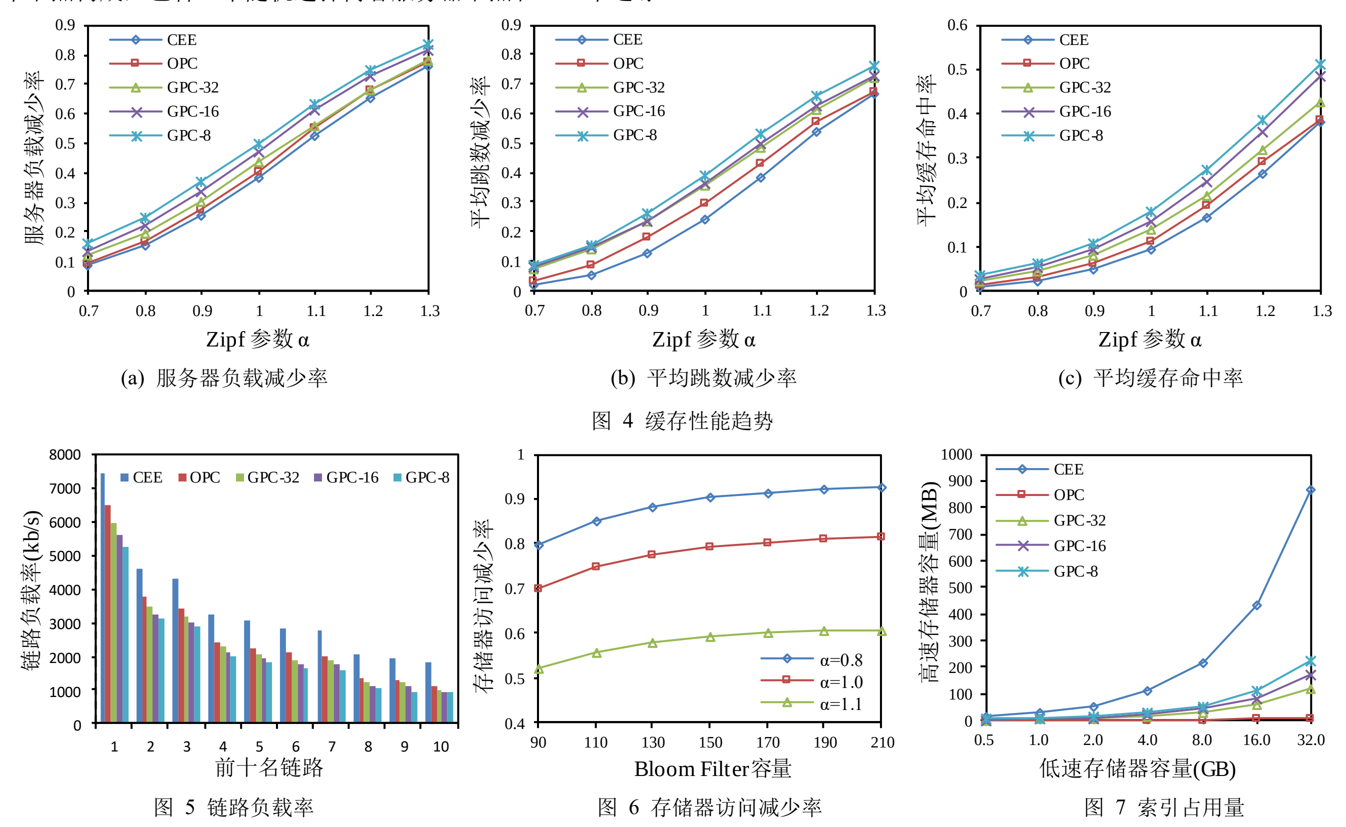
a)CEE (*cache everything everywhere*)[2]：每一个缓存设备都会将经过的任何数据包缓存下来。

b)OPC (*object-oriented packet caching*)错误!未找到引用源。：面向内容对象的包缓存策略，用以对象级别建立索引，并对一个内容对象的前 N 个数据包进行连续存储。

在实验中，假设用户请求服从 Zipf 分布错误!未找到引用源。，并且将默认的 Zipf 分布参数值设定为 1.0。此外，为了考察 Zipf 分布参数对缓存性能的影响，将其值的变化范围设定为 0.7~1.3。假定组成一个内容对象的数据包数量服从 *N（1000,250）*。通过实际采样得到的数据包的数量值分布于区间[68, 2026]。利用流量生成器来生成可变流行度的数据包并分别采用 BA 模型错误!未找到引用源。和 WS 模型错误!未找到引用源。构建实验拓扑。实验拓扑由 100 个节点构成，包含一个随机选择内容服务器节点和 66 个边缘路由器节点，其余节点为中间路由器节点。设定每个节点的缓存容量为 1000 个数据包，Bloom Filter 的默认容量为 150 Byte，并且将其变化范围设定为 90~210 Bytes 来考察其容量对过滤效果的影响。请求速率设置为 200 个请求/s。除了 Zipf 分布参数、分块数量和 Bloom Filter 容量外，其余参数都固定为默认值，表 1 描述了实验的主要参数设置。

表 1　实验参数设定

参数	默认值	变化范围
Zip 分布参数 α	1.0	0.7~1.3
分块数量	1 000	68~2 026
BF 容量	150 Bytes	90~210 Bytes
内容总数	1 000	-
缓存容量	1 000	-
请求速率	200 req/s	-
总节点数	100	-
边缘路由器数	66	-
仿真时间	180s	-



(a) 服务器负载减少率　(b) 平均跳数减少率　(c) 平均缓存命中率

图 4 缓存性能趋势

图 5 链路负载率　图 6 存储器访问减少率　图 7 索引占用量

4.2　性能指标

本文定义了如下性能指标来评估 GPC 方案的实际性能：

a)服务器负载减少率,用来描述缓存系统使内容服务器工作负载降低的比率，具体定义为

$$SLRR = 1 - \frac{S_counts}{R_counts} \tag{3}$$

其中:*S_counts* 表示由服务器响应的请求总数，而 *R_counts* 表示总的用户请求数。

b)平均跳数减少率,反映由于缓存系统的加入，使用户请求内容跳数减少的比率，定义为：

$$AHRR = 1 - \frac{\sum_{1}^{R_counts} hops_c}{\sum_{1}^{R_counts} hops_nc} \tag{4}$$

其中:$hops_c$ 表示缓存命中节点到请求用户之间的跳数,而 $hops_nc$ 表示请求用户到内容服务器之间的跳数,R_counts 表示总的用户请求数。.

c)平均缓存命中率,反映所有缓存节点的平均命中率，定义为.

$$ACHR = \frac{\sum_{R} Hit_counts_r}{\sum_{R} R_counts_r} \tag{5}$$

其中:Hit_counts_r 表示用户请求在路由器 r 处命中的次数，R_counts_r 表示路由器 r 收到的用户请求的总次数。

d)链路负载率,反映特定链路在一定时期内的流量负载状况。

e)存储器访问减少率,反映由于加入 Bloom Filter 使得访问存储器次数减少的比率，具体定义为

$$SARR = \frac{\sum_{R} F_counts_r}{\sum_{R} R_counts_r} \tag{6}$$

其中:F_counts_r 表示用户请求在路由器 r 处被 Bloom Filter 过滤的总次数，R_counts_r 表示路由器 r 收到的用户请求的总次数。

f)索引占用量,反映特定节点中缓存索引所占用的高速存储器的容量。

4.3 结果分析

在多种拓扑下进行大量的仿真实验,分析了各种策略下，Zipf 分布参数 α 变化（0.7~1.3）对缓存性能的影响。结果表明 GPC 策略在定义的评价指标上明显优于其他方案，同时内存的消耗量也大大降低了。

4.3.1 缓存性能

图 4 (a)~(c)分别显示了 Zipf 参数对缓存性能的影响。随着 α 的增加，所定义三个性能指标都有所改善。这是因为随着 zipf 分布参数的增大，热门内容的热度也随之增加，这有助于缓存系统的性能的发挥。如图 4 所示，本文方案在三个性能指标方面都优于其对比方案。另一方面，可以发现 GPC-8 获得了最优的缓存性能。这是因为分组大小越小，获得的热度值就越精确，从而作出的缓存决策也更加准确。与 OPC 相比，GPC-8 在三项指标上的平均改进分别为 29.53％，48.86％和 82.36％。而与 CEE 相比，性能指标的平均改善可以达到 37.65％，105.99％和 125.19％。

4.3.2 链路负载率

通过收集和分析链路数据传输状态来评估缓存策略对链路负载率的影响。图 5 显示了不同缓存策略下的前 10 名流量负载的链路的流量负载率。可以发现，使用 GPC -8 策略的场景下链路负载率明显低于其他策略。就该项指标而言，GPC -8 相对于 CEE 和 OPC 的平均改进为 40.45％和 19.08％。

4.3.3 外部存储器访问

图 6 显示了 Bloom Filter 的作用。通过 Bloom Filter，大多数请求都直接转发而无须访问外部存储器。在图 6 中，随着 Bloom Filter 容量的增加，转发效果逐步提升。但同时可以发现性能曲线是上凸的,表明性能改善率不断下降。也就是说 Bloom Filter 的效果具有上限,因而不能为了获得更好地过滤效果而一味增大其容量。

4.3.4 索引占用量

图 7 显示了低速存储器容量和索引大小之间的关系。假设每个索引条目的大小是 40 个字节,而数据包大小是 1500 Byte。GPC-8，GPC-16，GPC-32 的 Bit-Map 大小分别为 1，2，4 字节。对于 GPC-8 来说一个索引条目最多可以表示 8 个数据包，而一个 CEE 索引条目只能标识一个数据包。因此，通过增加组大小，索引条目数量也可以减少，因而高速内存使用量可以相应减少。

5 结束语

本文提出了一种全新的数据包级别的缓存优化方法，它从多个角度入手缓解了 NDN 架构下数据包级别缓存所遇到的扩展性问题。该策略将组成内容对象的若干数据包按照顺寻分成多个分组，并使用分组前缀建立索引条目以减少高速内存的使用量。另一方面，利用外置的 Bloom Filter 过滤无效的缓存查询，加速数据包的转发速率。此外，该方案根据分组级别的热度统计来优化缓存决策，提高缓存整体性能。实验结果表明，提出的方案在服务器负载减少率，平均跳数减少率和平均缓存命中率等方面优于现有解决方案,并且显着提高了资源利用率。下一步将继续完善该方案，并将其部署在在实际的 NDN 平台中，进行进一步测试和应用。

参考文献：

[1] Cisco. Cisco visual networking index: forecast and methodology, 2016–2021 [EB/OL]. (2017) [2017-07-15]. https://www. cisco. com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360. html

[2] Wang Lan, Bayhan S, Kangasharju J. Optimal chunking and partial caching in information-centric networks [J]. Elsevier Science Publishers B. V. 2015, 61 (C): 48-57.

[3] Jacobson V, Smetters D K, Thornton J D, *et al.* Networking named content [C]// Proc. of ACM CoNEXT. 2009: 1-12.

[4] Koponen T, Chawla M, Chun B, *et al.* A data-oriented (and beyond) network

architecture [C]// Proc. of ACM SIGCOMM. 2007: 181–192.

[5] FP7 SAIL project [EB/OL]. http://www. sail-project. eu/

[6] Fotiou N, Nikander P, Trossen D, *et al.* Developing information networking further: from PSIRP to PURSUIT [C]// Proc of International Conference on Broadband Communications, Networks and Systems. Berlin: Springer, 2010: 1-13.

[7] Chai W K, He Diliang, Psaras I, *et al.* Cache less for more in information centric networks [C]// Proc of NETWORKING. 2012: 27-40.

[8] Psaras I, Chai W K, Pavlou G. Probabilistic in-network caching for information-centric networks [C]// Proc of ACM SIGCOMM Workshop on ICN. New York: ACM Press, 2012: 55-60.

[9] Zhang Meng, Luo Hongbin, Zhang Hongke. A survey of caching mechanisms in information-centric networking [J]. IEEE Communications Surveys & Tutorials, 2015, 17 (3): 1473–1499.

[10] Bernardini C, Silverston T, Festor O. MPC: popularity-based caching strategy for content centric networks [C]// Proc. of IEEE International Conference on Communications. 2013: 3619-3623.

[11] Wang Wei, Sun Yi, Guo Yang, *et al.* CRCache: exploiting the correlation between content popularity and network topology information for ICN caching [C]// Proc. of IEEE International Conference on Communications. 2014: 3191-3196.

[12] Ren Jing, Qi Wen, Westphal C, *et al.* MAGIC: a distributed max-gain in-network caching strategy in information-centric networks [C]// Proc of IEEE INFOCOM. 2014: 470–475.

[13] Wu Haibo, Li Jun, Zhi Jiang. MBP: a max-benefit probability-based caching strategy in Information-Centric Networking [C]// Proc of IEEE International Conference on Communications. 2015: 5646-5651.

[14] Hu Xiaoyan, Gong Jian, Cheng Guang. Enhancing in-network caching by coupling cache placement, replacement and location [C]// Proc of IEEE International Conference on Communications. 2015: 5672-5678.

[15] Banerjee B, Seetharam A, Tellambura C. Greedy caching: a latency-aware caching strategy for information-centric networks [C]// Proc of IFIP Networking. 2017.

[16] Thomas Y, Xylomenos G, Tsilopoulos C, *et al.* Object-oriented packet caching for ICN [C]// Proc of International Conference on Information-Centric Networking. 2015: 89-98.

[17] Wang Lan, Hoque A, Yi Cheng, *et al.* OSPFN: an OSPF based routing protocol for named data networking [R]. University of Memphis and University of Arizona, 2012.

[18] Hoque A, Amin S, Alyyan A, *et al.* NLSR: named-data link state routing protocol [C]// Proc of ACM SIGCOMM Workshop on Information-Centric Networking. New York: ACM Press, 2013: 15-20.

[19] Afanasyev A, Moiseenko I, Zhang Lixia, ndnSIM: NDN simulator for NS-3, NDN-0005 [R]. NDN, 2012.

[20] Henderson T R, Roy S, Floyd S, *et al.* ns-3, project goals [C]// Proc of Workshop on ns-2: the IP Network Simulator. New York: ACM Press, 2006.

[21] Breslau L, Cao Pei, Fan Li, *et al.* Web caching and Zipf-like distributions: evidence and implications [C]// Proc of the 18th Joint Conference of the IEEE Computer and Communications Societies. Proceedings. 2002: 126-134.

[22] Barabási A, Albert R. Emergence of scaling in random networks [J]. Science, 1999, 286 (5439): 509-512.

[23] Watts D. J, Strogatz S. H. Collective dynamics of small world networks [J]. Letters to Nature, 1998, 393: 440-442.